# Refine Search

## Search Results -

| Terms | Documents |
|---|---|
| L15 same result | 20 |

**Database:**
- US Pre-Grant Publication Full-Text Database
- **US Patents Full-Text Database**
- US OCR Full-Text Database
- EPO Abstracts Database
- JPO Abstracts Database
- Derwent World Patents Index
- IBM Technical Disclosure Bulletins

**Search:** L17

[Refine Search]

[Recall Text ⬍]  [Clear]  [Interrupt]

---

## Search History

**DATE:** Thursday, December 08, 2005    Printable Copy    Create Case

| Set Name side by side | Query | Hit Count | Set Name result set |
|---|---|---|---|
| | *DB=USPT; PLUR=YES; OP=OR* | | |
| L17 | L15 same result | 20 | L17 |
| L16 | L15 same behavior | 2 | L16 |
| L15 | L14 same simulat$ | 53 | L15 |
| L14 | error adj2 event | 2417 | L14 |
| L13 | L12 and script | 0 | L13 |
| L12 | 6353898[pn] or 6182248[pn] or 6167479[pn] or 5841960[pn] or 5699502[pn] or 5671352[pn] 5475624[pn] or 5425036[pn] or 5146460[pn] or 5130988[pn] or 5008885[pn] or 4999837[pn] or 4759019[pn] | 13 | L12 |
| | *DB=USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=OR* | | |
| L11 | L9 same simulat$ | 4 | L11 |
| L10 | L9 and 11 | 0 | L10 |
| L9 | 17 same event | 114 | L9 |
| L8 | L7 and 11 | 1 | L8 |

| L7 | script adj1 file | 1415 | L7 |
|----|------------------|------|----|
| L6 | L5 same (fail$ or error) | 4 | L6 |
| L5 | L1 same result | 52 | L5 |
| L4 | L3 same event | 14 | L4 |
| L3 | L1 same response | 38 | L3 |
| L2 | l1 same report$ same error$ | 1 | L2 |
| L1 | behavior adj1 simulat$ | 364 | L1 |

END OF SEARCH HISTORY

☐    Generate Collection   Print


L6: Entry 1 of 4                      File: USPT              Jan 15, 2002


DOCUMENT-IDENTIFIER: US 6339837 B1
TITLE: Hybrid method for design verification


Detailed Description Text (21):
FIG. 2 illustrates a verification engine in the first embodiment, comprising four
steps: an interpreter 201, a cycle counter 202, an unroller 203, and an equivalence
checker 204. Interpreter 201 translates verification structure 111 from a hardware
description language into a data structure representing a network of gates and
registers. Cycle counter 202 determines the clock cycle number involved in the
behavior being simulated up to the ending indicator. Unroller 203 traverses the
network of gates and registers in the result of interpreter 201 from result flag
104, with the clock cycle number from cycle counter 202, to free variables 101 and
builds a data structure representing a network of gates. Unroller 203 copies the
contributing gates for each clock cycle and connects the consecutive copies by
replacing each register with a wire between its data input and its data output. For
the initial cycle, each register's data output takes the register's initial value
if there is any, or otherwise one of free variables 101. For the final cycle, the
register inputs are ignored if they are not connected to result flag 104.
Equivalence checker 204 determines whether the network of gates makes every copy of
result flag 104 generated by unroller 203 equivalent to the predetermined value of
result flag 104. The answer from equivalence checker 204 is taken to a decision
point 205, which declares the verification a success 206 if it is all positive.
Otherwise it declares the verification a failure 207. All steps before taking the
answer from equivalence checker 204 can include Boolean simplification techniques
for the sake of reducing the amount of computation.

S, 146/460

☐ ▓▓▓ Generate Collection ▓▓▓ | Print |

L4: Entry 2 of 14                      File: USPT                  Nov 30, 2004

DOCUMENT-IDENTIFIER: US 6826518 B1
TITLE: Method for distributed agent-based non-expert simulation of manufacturing
process behavior

Brief Summary Text (21):
From the agent perspective, a method for distributed agent-based simulation of
manufacturing process behavior, the simulation having a plurality of agents
corresponding to individual processes forming a manufacturing technique, comprises
the steps of: receiving a message from an agent; identifying in the received
message a discrete event selected from the group consisting of a clock tick event,
a resources received event, and a request for output production event; causing an
associated process to perform an activity in response to the identified event; and,
messaging an adjacent agent in response to the identified event.

CLAIMS:

7. A method for distributed agent-based simulation of manufacturing process
behavior, the simulation having a plurality of agents corresponding to individual
processes forming a manufacturing technique, the method comprising the steps of:
receiving a message from an agent associated with one of a plurality of different
manufacturing techniques comprising a pull, a push, or a takt manufacturing
technique, wherein the agent is responsive to a discrete event selected from the
group consisting of a clock tick message, a resources received message, and a
request for output production message; identifying in said received message a
discrete event selected from the group consisting of a clock tick event, a
resources received event, and a request for output production event; causing an
associated process to perform an activity in response to said identified event;
and, messaging an adjacent agent in response to said identified event.

☐  Generate Collection   | Print|


L17: Entry 9 of 20                          File: USPT              Apr 18, 2000


DOCUMENT-IDENTIFIER: US 6052524 A
TITLE: System and method for simulation of integrated hardware and software
components


Brief Summary Text (14):
There have been attempts to overcome some of the timing problems associated with
simulators and emulators using hardware verification languages including: "Logic
Simulation Using a Hardware Accelerator Together with an Automated Error Event
Isolation and Trace Facility," U.S. Pat. No. 5,146,460, by Ackerman et al; "Timing
Analysis for Logic Optimization Using Target Library Delay Value," U.S. Pat. No.
5,475,605, by Lin; and "Hardware-Software Debugger Using Simulation Speed Enhancing
Techniques Including Skipping Unnecessary Bus Cycles, Avoiding Instruction Fetch
Simulation, Eliminating the Need for Explicit Clock Pulse Generation and Caching
Results of Instruction Decoding," U.S. Pat. No. 5,678,028, by Bershteyn, et al.
However, as electronic devices continually increase in speed and require different
types of high speed and low speed memory, such attempts to improve the timing
problems still do not provide adequate simulation of timing interactions between
hardware and software components in an electronic device.

☐    Generate Collection   Print

L17: Entry 18 of 20                         File: USPT                    Sep 8, 1992


DOCUMENT-IDENTIFIER: US 5146460 A
TITLE: Logic simulation using a hardware accelerator together with an automated
error event isolation and trace facility

Abstract Text (1):
Software simulators of logic design circuits run slowly but are capable of
providing very finely detailed error trace analyses. On the other hand, hardware
accelerators operating to perform similar functions are very fast in their
execution but are not capable of practically isolating error states or other
critical conditions. Accordingly, the present invention provides an interactive
system combining software simulators and hardware accelerators so that when desired
test results do not favorably compare with simulated results, a mechanism is
provided for storing the current hardware accelerator state and restoring the
accelerator to a previous checkpoint state which has been saved as a result of a
prior periodic interruption. The hardware accelerator is then operated for a time
sufficient to bring it up to a state that occurs just before the detected
miscomparison. At this point, state information from the hardware accelerator is
supplied to a software simulator for detailed error analysis and fault tracing. The
hardware accelerator may then be restarted where it left off or with a different
task. In this way, optimal utilization is made of expensive hardware accelerator
resources while nonetheless providing error event isolation and the ability to
generate detailed traces of the simulated behavior.

☐ ▓▓▓▓▓▓▓ Generate Collection ▓▓▓▓▓▓▓ � Print ▓

L8: Entry 1 of 1                        File: USPT              Aug 22, 2000

DOCUMENT-IDENTIFIER: US 6106298 A
TITLE: Reconfigurable easily deployable simulator

Abstract Text (1):
A modular, reconfigurable and easily deployable simulator system includes a
plurality of vehicle personality kits and at least one frame to support a single
crewstation. Members of a selected kit are coupled to the frame. A multi-platform
control system provides on-the-window and in vehicle instrument displays in real-
time in response to the crewperson manipulating platform controls. An editor with a
graphical user interface provides a capability to modify program segments of the
control system thereby altering the behavior of the simulated platform.

Detailed Description Text (75):
The configuration file specifies the segments and interconnecting messages which
ultimately carry out the control process leading to behavior which simulates the
selected platform. A configuration file includes a manager segment. The manager
segment issues a loud configuration call. The configuration file is then read in
the segment executable so called. The called segments in turn issue enter
configuration calls to register with the CSL 16. They then initiate various generic
procedure calls to access message data. When the segments are all initialized, the
simulation process can begin.

Detailed Description Text (85):
A particular processor can be specified to which a given segment will be
associated. Absent binding a segment to a processor, CSL 16 and the operating
system may move a particular segment to a different host for execution. A path
parameter specifies a UNIX path for the segment executable or script file that will
run the segment executable. A priority parameter controls the priority that the
segment will receive relative to other processors or segments being run on the
host. A lock parameter makes it possible to lock a segment's text and data area
thereby making it immune to routine swapping by the operating system. A list of
message names must be provided along with the type of access to indicate those
messages that the segment needs access to. Finally, X and Y position parameters for
the configuration editor graphical user interface must also be specified.

☐ ▓▓▓ Generate Collection ▓▓▓  | Print |

L11: Entry 2 of 4                      File: USPT              Mar 27, 2001

DOCUMENT-IDENTIFIER: US 6208955 B1
TITLE: Distributed maintenance system based on causal networks

Detailed Description Text (35):
In step 610 detailed simulation for model testing and validation is performed. The
causal network diagnostic model operates on a static-input-variable-state snapshot
to produce a diagnostic output. This process can be performed either through a
manual mode or by executing simulation script files that specify the timing
constraints for individual modeling components. In the manual mode, failure sources
can be selected to change their states so as to simulate the rippling effect of a
failure source through avionics system 100. Thus, system level evaluation for
multiple system states and diagnostic outputs is possible in a short period of
time. For the simulation script method, an input/output (I/O) shell simulator is
used to simulate inputs into the model domain and control the timing relationships.
It is the simulator shell's job to activate the diagnostic process when an
appropriate system state has been reached. Activation of the diagnostics is
therefore event-driven.